一种基于沙漏网络的多层次协同搜索方法 *

陈桂荣，邱仲禹，粟　涛†，陈弟虎

(中山大学 电子与信息工程学院，广州 510006)

摘　要：目前，人工智能快速发展，人们可以通过探索硬件设计空间使优秀的神经网络算法高效部署在 FPGA 加速器。然而，由于参数量大、操作过于复杂而导致算法与硬件难以匹配，加速效率不高。为了算法和硬件二者匹配性更强，提出了一种多层次协同搜索的方法，采用 SPOS 搜索策略并以检测准确率和延时为评估目标，搜索出最优神经网络架构、量化方式和硬件设计参数组合。该方法应用在姿态识别中具有优异性能的沙漏网络中，在获取候选子网络量化前、后的检测准确率的同时对硬件设计参数使用遍历搜索得到预估延时，根据目标函数获取最高得分的最优组合。为了保证获取的数据有效性，子网络需要进行重新训练、量化后重新推理得到检测准确率，获取硬件设计参数则利用基于 Spinal HDL 设计的加速器模板进行仿真测试得到测试延时。就平均而言，该方法比文献[1]减少了 83.3%的参数，准确率只下降了 0.69；比传统加速方法平均减少了 33.2%参数量，准确率只下降了 0.46，网络推理的测试总延时减少 22.1%，在沙漏块的测试延时减少 67.8%。总体而言，该协同搜索方法对于沙漏网络的优化有一定效果，比传统加速设计方法更有优势。

关键词：神经网络；FPGA；协同搜索；沙漏网络；延时模型

中图分类号：TP391　　doi: 10.19734/j.issn.1001-3695.2022.01.0018

Multi-level co-exploration method based on hourglass network

Chen Guirong, Qiu Zhongyu, Su Tao†, Chen Dihu

(*School of Electronics & Information Technology*, *Sun Yat-Sen University*, *Guangzhou* 510006, *China*)

Abstract: At present, with the rapid development of AI, people can efficiently deploy excellent neural network algorithms on FPGA accelerators by exploring the hardware design space. However, due to the large amount of parameters and complex operation, it is difficult to match the algorithm with the hardware, and the acceleration efficiency is low. In order to better match the algorithm and hardware, this paper proposed a multi-level co-exploration method, adopted SPOS search strategy, aimed at accuracy and latency, to obtain the optimal neural network architecture, quantization method and hardware design combination. It applied the method to hourglass network which have high accuracy in pose estimation. While obtained the accuracy before and after quantization of candidate sub networks, it used traversal method to search hardware design parameters and obtain the estimated latency, and then got the optimal combination with the highest score according to the target function. In order to ensure the effectiveness of the obtained data, it retrained the sub network, then quantified and inferenced again to obtain the accuracy. It simulated the obtained hardware design parameters to get the testing latency, using the accelerator template designed based on Spinal HDL. On average, Co-exploration method reduced the parameters by 83.3% and with only 0.69 accuracy loss compared with the original structure; reduced the parameters by 33.2%, with only 0.46 accuracy loss, reduced the total testing latency of network inference by 22.1% and reduced the testing latency in hourglass block by 67.8% compared with the traditional acceleration method. Overall, the co-exploration method in this paper has a certain effect on the optimization of hourglass network, and it has more advantages than the traditional acceleration method.

Key words: neural network; FPGA; co-exploration; hourglass network; latency model

0　引言

随着人工智能快速发展，姿态估计是计算机视觉领域的主要研究方向之一，在很多领域有着重要研究价值和应用前景。堆叠式沙漏网络[1]在二维姿态估计算法中，表现优异，目前很多算法都是基于沙漏网络的变体[2,3]。但是设计者的初衷更多是考虑其检测准确率，而该网络模型复杂、参数量大在边缘计算加速推理方面有很大的挑战。目前在边缘加速方面，大都是预先有算法网络结构，然后基于该固定结构进行加速推理，却缺乏对于算法、硬件等多方面协同设计的方法，因此存在高准确率算法却难以匹配到硬件进行加速推理的问题[4,5]，所以在加速计算方面仍有改进的空间。

在二维姿态识别领域中，存在很多较为优秀的算法[1,6~9]。以 PCKh@0.5 的值作为检测准确率衡量指标，文献[1]的沙漏网络检测准确率是 90.9，但是其参数量有 25.1M；文献[7]的 SimpleBaselines 准确率是 91.5，虽然采用简单有效的结构，但是其参数量高达 68.6M；文献[8]的 HR Net 其检测准确率是 92.3，但是其参数量也有 28.5M 之多，而且参数均为浮点数。这些优秀的算法无论是在整体网络的实现过程还是参数量，在边缘加速的实现过程都是巨大的挑战，难以匹配到硬件进行加速。

近年来，科研团队在神经网络算法、压缩量化、硬件设计等方面的性能提升作出很大的贡献。首先为了实现提高神经网络的性能这一目标，很多研究人员采用神经网络架构搜

收稿日期：2022-01-21；**修回日期：**2022-03-11　**基金项目：**广东省重大科技计划项目(2021B110127007，2019B010140002)

作者简介：陈桂荣(1997-)，男，广东肇庆人，硕士研究生，主要研究方向为 AI 加速、人工智能；邱仲禹(1996-)，男，广东佛山人，硕士研究生，主要研究方向为 AI 加速、人工智能；粟涛(1977-)，男(通信作者)，湖南益阳人，副教授，博士，主要研究方向为人工智能，芯片与应用系统设计(sutao@mail.sysu.edu.cn)；陈弟虎(1963-)，男，四川绵阳人，教授，博士，主要研究方向为集成电路设计方法，深度学习与图像识别技术.

索[10~13](neural network architecture search, NAS)这种先进的方法，从人工调参的繁琐工作中脱离出来。文献[11]提出了 NAS Net，设计一个新颖的搜索空间，并使用强化学习优化方法提高了搜索性能。文献[12]引入 ENAS，利用权重参数共享方法提升了 NAS 的效率，GPU 运算时间缩短了 1000 倍以上。文献[14]提出了 One-Shot 的方法，通过训练一个超网络作为辅助模型，整体搜索速度很快。其次，为了减少神经网络的参数，加快神经网络推理，不少科研人员在量化方面取得很好的进展[15]。文献[16]将网络权值二值化让计算主要在正 1 或负 1 间进行，降低了网络大小和计算量。文献[17]将推断过程中的浮点数运算量化为整数运算，最终将权重和激活函数量化为 8 位。文献[18]采用混合精度量化方式以找到合适的量化位宽，实现降低能耗的目标。为了能更快的实现神经网络推理，在加速器硬件设计中有很多地方值得关注，如处理单元(Process element，PE)数目、并行度、数据流复用方式等等，目前设计人员在这些方面做了很多探索。文献[19]利用 Roofline 模型对卷积神经网络(Convolution Neural Network，CNN)加速器数据流技术的设计空间探索，探索不同数据流复用方式的性能。文献[20]在探索了 PE 的利用率和并行度，实现了性能的大幅度提升。文献[21]提出了一种加速器架构可以进行卷积层间流水，在输出特征图数、宽度和高度三个维度进行展开实现并行化计算的。然而这些工作都是从单一角度出发，例如从算法准确率、硬件参数量、硬件设计方法等单一优化，并未协同考虑进行共同优化，以便获取最优性能。

在多目标协同搜索和硬件感知搜索[4,5,18,22~30]相关研究越来越多，设计者不仅考虑算法层面的准确率而且考虑很多硬件性能，如功耗、耗时、面积、资源利用等等。文献[4]考虑了层和数据流复用模式之间的适应性，选择最合适的数据流复用模式，高效部署所搜索的网络。文献[24]利用强化学习方法搜索，搜索空间包括算法架构、混合精度量化方式等，并建立延时模型和能耗约束，实现能耗和延时的下降。文献[29]定义了一个新的硬件搜索空间，在目标 FPGA 平台上生成具有延时保证的最优神经网络模型。文献[30]等提出了一种器件-电路-架构的协同探索框架，包括器件类型、电路拓扑结构和神经网络超参数等实现性能的优化。但是这些工作基本都是基于图像分类，而对于姿态识别、沙漏网络却缺乏该方面的研究工作。

综上所述，在姿态识别方面，优异的算法缺乏硬件的适配性；在网络优化、压缩、硬件加速方面却更多的考虑单一性能优化；在多目标优化的协同方法中缺乏对姿态识别方面的研究工作。而神经网络架构搜索方法可以自动探索神经网络的设计优化空间，设计者可以更进一步改变网络结构实现更高精度。与此同时，设计者还可以将其他硬件端的目标作为优化对象。这些方法很好的解决了高准确率算法与硬件难以匹配的问题，可以以很低的精度损失实现更简便的方式进行高速推理。

本文兼顾检测准确率和延时提出了一种面向姿态识别的沙漏网络多层次协同搜索方法，充分探索算法和硬件设计等方面的优化空间，主要的贡献有：针对沙漏网络的结构，提出了算法、压缩量化、硬件设计等方面的协同优化方法，以低准确率损失实现了性能较大的提升；.建立了一个接近加速器真实运行的延时模型进行分析指导协同搜索过程；为了对延时模型的准确性进行验证，利用 Spinal HDL 设计了加速器模板对获取的数据流、并行度硬件设计参数进行仿真测试。

1 多层次协同搜索方法

1.1 问题定义

在传统的神经网络到硬件加速部署设计过程中，算法设计者关注网络模型的准确率，因此各种 NAS 工作的目标是寻找一个精度更高的网络结构，然而硬件设计师更多地关注加速器的性能，如延时、功耗等，所以很容易出现算法和硬件不匹配的问题。多层次协同方法旨在优化沙漏网络的推理延时，同时保证准确率损失最小。

准确率的获取需要候选模型对验证集进行推理，本文所有的准确率都是 MPII 数据集特定指标值 PCKh@0.5。在利用 MPII 数据集评估时，通常利用关节点正确定位百分比(percentage of correct keypoints，PCK)评估人体关节点定位的准确率，通常记为 PCK@k。当关键点 i 的预测位置与真实位置间的距离 d_i 小于头部长度 L_{haed} 一定比例 k，称为 PCKh@k，计算公式如下。

$$\text{PCKh}=\frac{\sum_i \delta\left(d_i<kL_{\text{head}}\right)\delta\left(v_i>0\right)}{\sum_i \delta\left(v_i>0\right)} \tag{1}$$

当 k 的值为 0.5 时，即表示为 PCKh@0.5。

而网络推理延时的获取是使用数学建模来预测，延时可以用以下式(2)来描述，延时模型后续会专门进行介绍。

$$L_{total}=L_{calc}+L_{on-chip}+L_{off-chip} \tag{2}$$

总延时 L_{total} 包括计算延时 L_{calc} ,片上数据移动延时 $L_{on-chip}$ ，片外内存访问延时 $L_{off-chip}$.

为了更好评估搜索的网络模型和加速器的设计参数，本文提出一个目标函数，同时目标函数可以用于指导整个协同搜索过程，如式(3)所示

$$\text{Score}=(\alpha*\text{Acc}+\beta*\text{Q_Acc}+\gamma*L_{total}) \tag{3}$$

Acc ,Q_Acc 为候选模型量化前、量化后的准确率,二者主要是保证量化前后都有较好的准确率。这里准确率和延时进行归一化处理，保证 Score 在 0-100 的范围。α,β,γ 为比例因子，根据多次调参后获得合适的比例，本文采用的比例分别为 0.4,0.4,0.2。

1.2 方法概述

多层次协同搜索方法框架主要包含超网络训练、协同搜索和硬件部署等部分，如图 1 所示。SPOS[15](Single Path One Shot)将超网络训练和搜索分开进行，搜索过程是采用遗传算法进行多次迭代。协同搜索方法的输入约束主要是在算法端的沙漏块的模型、中间压缩的量化方式以及加速器设计的数据流复用方式和并行度，其中并行度会以 FPGA 的综合实现结果的 DSP 数目反映出来。搜索的输出是目标函数的评估得出最优的网络模型、量化方式、加速器数据流和并行度参数。网络架构需要进行重新训练、量化、推理，以保证该网络结构的有效性和准确性。而加速器设计参数则会在加速器模板进行修改参数以获取 RTL(Register transfer level)代码，最终目标是在 FPGA 实现高速推理。

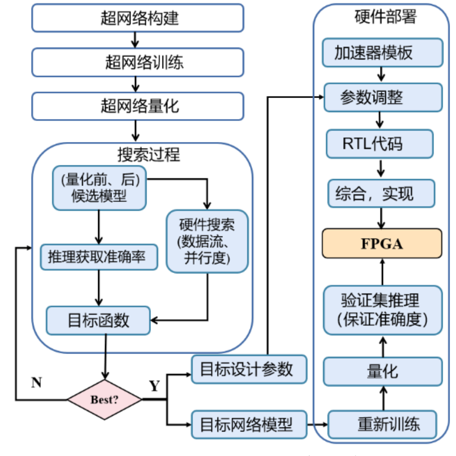


图 1 多层次协同搜索方法

Fig. 1 Multi-level co-exploration method

整个搜索过程的实现是先对超网络的沙漏块进行编码，方便使用遗传算法对沙漏块结构完成交叉、变异等遗传操作，再对超网络训练、量化，然后进行搜索；在搜索阶段，根据沙漏块的编码随机采样获得候选子网络作为初始种群，将量化前的候选子网络在验证集进行推理，同时将使用不同量化方式的子网络进行推理，获取量化前、后的准确率以便返回到目标函数进行评估，在候选子网络中最优的子网络会被遗传算法选取出来作为遗传的对象进行下一轮的迭代。在验证集验证的同时，硬件设计会采用遍历的方式进行搜索，将延时模型评估获得最低延时返回到目标函数同时获得相应的硬件设计参数。通过将量化前、后的准确率和预估延时，目标函数会获得相应的得分以便确保该子网络结构、量化方式、硬件设计参数均为最优。

由于超网络训练时采用权值共享的方式，子网络的准确率还有提升的空间，所以当获得了目标的网络模型，会对其结构进行重新训练、量化、验证等，以保证获取更好的准确率。获取的硬件设计参数，会调整加速器模板的参数，获取新的 RTL 代码之后进行综合、实现等后续过程。

2　多层次协同搜索空间定义

2.1　算法层面

原始的沙漏块结构如图 2 所示，每一个小块都是残差块，残差块操作在硬件的实现较为复杂，并且沙漏块的每一条支路特征融合都需要额外的缓存，因此在支路的特征融合方面和特征提取的操作存在优化的空间。

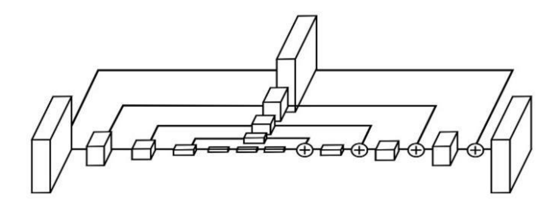


图 2　沙漏块的原始结构[1]

Fig. 2　Original structure of hourglass block[1]

在支路的特征融合方面，本文的搜索空间有三种选择，第一种是无特征融合，即去除旁路分支，不保留以前的特征，第二种是无卷积的直接融合，即没有任何卷积开销的快捷方式，第三种是原来的卷积特征融合，如图 3 所示。无卷积特征融合比卷积特征融合减少很多的缓存、计算时间和额外的权重参数。无特征融合减少缓存时间最多。在操作算子块搜索空间方面，原来的残差块需要很多的计算和缓存时间，本文尝试用深度可分离卷积块替换以减少计算和缓存的开销。

为了方便对优异的沙漏块结构进行遗传，这里对沙漏块在加速器的计算顺序进行编码，融合方式分别是 0,1,2，操作算子块分别是 0,1，例如原始的卷积特征融合其对应为 2，在原始的残差操作块编码为 0，所以原始结构的编码为[2,0,2,0,2,0,2,0,0,0,0,0,0]，一共 13 个位置编码，而图 3 则表示[2,0,1,0,0,0,2,0,0,0,0,0,0]。

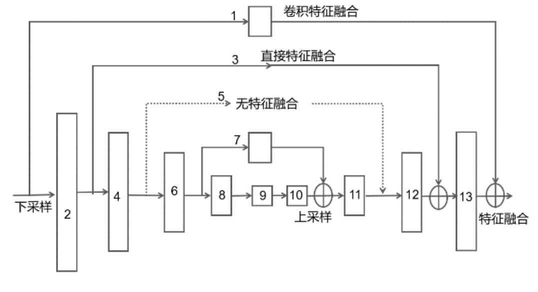


图 3.特征融合搜索空间

Fig. 3　Search space of feature fusion

2.2　量化层面

在神经网络压缩方面，有量化、剪枝、稀疏化、蒸馏等方式，其中量化是最为常用的。因为神经网络每一层的数据分布都不一样，每一种量化方式有一定的差异，所以使用不同量化方式对不同的结构进行量化可能出现不同的结果。为了探索最好的量化方式，本文量化的探索空间是定点和整型。对于 8 位定点型量化，本文采用式(4)对整数位的位宽分别是 1、2、3 位进行探索，其中 m 表示有符号二进制补码，b 表示小数位宽，N 表示量化的总位宽。

$$m=(1/2^b)\left[-2^{N-1}x_{N-1}+\sum_{n=0}^{N-2}2^n x_n\right] \tag{4}$$

例如 Fix8(2)表示整数位有 2 位，1 位为符号位，其余的是小数位，如图 4 所示。

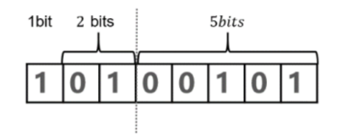


图 4　Fix8(2)的定点数表示

Fig. 4　The fixed point representment of Fix8(2)

另一种方式是 8 位整型的方式进行探索，主要为如下式(5)(6)所示。

$$f=S(q-Z) \tag{5}$$

$$q=\text{round}\left(\frac{f}{S}+Z\right) \tag{6}$$

其中：f 为浮点数，S 表示缩放系数，q 为量化后整数 ，Z 表示零点。

2.3　硬件设计层面

CNN 加速器设计需要根据 FPGA 的资源情况设计较为合理的数据流复用方式和并行度，因为数据流和并行度都会影响加速器性能，如果数据流不适合可能造成额外的延时开销，如果并行度过大可能资源不够，并行度过小造成资源浪费。

1) 数据复用方式

数据流是指卷积计算的顺序，主要分为输出复用(Output Reuse, OR)、输入复用(Input Reuse, IR)和权值复用(Weight Reuse, WR)[20]，不同数据流在加速器表现为计算的顺序不同，每一种计算的顺序有不同的运行时间。因此，在加速器设计需要根据 FPGA 的资源情况确定其最优的数据流。其中输出复用伪代码如下算法 1 所示，相应的符号注解如表 1 所示。

表 1　部分符号解释

Tab. 1　Explanation of Some notations

符号	解释
Ic, Oc、Oh、Ow	输入通道，输出通道、长、宽
Tic,Toc、Tox、Toy、Kh,Kw	输入通道，输出通道、长、宽分块、卷积核长、宽
Pic ,Poc，Pox，Poy	输入通道，输出通道、长、宽并行度、
x,y,m,n 等	循环变量

算法 1：　输出数据流复用(OR)

```
输入：IFM[Ic][iw][ih], Weight[Ic][Oc][Kw][Kh]
输出：OFM[Oc][Oh][Ow]
for(y=0;y<Oh;y+=Toh){
 for(x=0;x<Ow;x+=Tox){
  for(m=0;m<Oc;m+=Toc){
   for(n=0;n<Ic;n+=Tic){
   //从 DDR 读数据
```

2)并行度

并行度是指在卷积计算时可以进行多少个并行的计算，

即表示同时计算多少个乘法。并行计算的方式可以在不同的维度展开，例如卷积核、输入输出通道，特征图的长宽等等。在上述三种数据流下，Pic、Poc 分别对应于输入和输出通道的并行计算，Pox、Poy 则表示卷积窗口的并行计算，即输出特征图的长、宽方向的特征点，不同的并行度组合也会表现出不一样的计算时间。

如图 5 所示，表示输入通道的并行计算，即在完成输出特征图的计算过程中，需要对卷积核与输入特征图对应通道相乘并累加后得到一个输出特征点，而该过程在 FPGA 上实现是利用多个乘法器并行计算完成的,因此并行度高其计算延时可能就少。其他维度的并行都是基于卷积计算的原理进行，这里不再赘述。

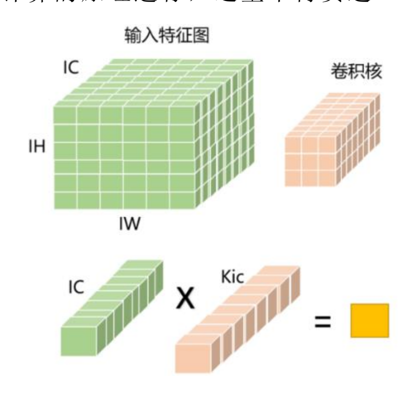


图 5　输入通道并行计算

Fig. 5　The parallel computing of input channel

2.4　多层次协同搜索空间

经过上面的分析，这里总结了本文多层次协同搜索的空间，如下表 2 所示。

表 2　多层次协同搜索空间

Tab. 2　Search space of multi-level co-exploration

变量		算法端搜索空间		
		量化方法搜索空间	硬件设计搜索空间	
算子块	残差卷积块	Int8	数据流	总并行度
	深度可分离卷积块	Fix8(1)	IR	
	无特征融合	Fix8(2)	OR	<=1600
支路融合操作	直接特征融合	Fix8(3)	WR	
	卷积特征融合	----	---	

3　延时模型

为了准确的评估协同搜索的结果好坏，本文建立了一个接近加速器真实运行的延时模型，如上面式(2)。延时模型充分考虑了计算、片上和片外数据访问的时间成本，比简单地用网络模型参数和计算量估计延时更加准确和全面。在 L_{calc} 的建模中，本文考虑了由于 PE 利用而导致的实际计算性能低于峰值计算性能的问题；在 $L_{off-chip}$ 的建模中，充分考虑了不同数据流下的重复数据访问模式。该模型适用于描述具有不同资源约束、数据流和并行性的普通 CNN 加速器的延时估计。

3.1　计算延时

计算延时表示计算整个网络所需要的时间，其中卷积占网络计算的 90%以上。计算延时可以表示为计算量除以总并行度和利用率的乘积，如式(7)所示，其中普通卷积的计算量如式(8)所示。PE 的利用率可以表示为式(9)。

$$\boldsymbol{L}_{calc}=\frac{\text{总计算量}}{\text{并行度}\ \times \text{PE 利用率}} \tag{7}$$

$$L_{calc}=\sum_{layer}\frac{O_w\times O_h\times I_c\times O_c\times K_w\times K_w}{P_{ox}\times P_{oy}\times P_{ic}\times P_{oc}\times PE_Utility} \tag{8}$$

$$PE_U tility=\frac{PE\ \text{有效计算量}}{\text{PE总计算量}}= \tag{9}$$

$$\frac{O_w\times O_h\times I_c\times O_c\times K_w\times K_h}{(O_w+O_w\%P_{ox})\times(O_h+O_h\%P_{oy})\times(I_c+I_c\%P_{ic})\times(O_c+O_c\%P_{oc})}$$

3.2　片上访存延时

片上访存延时是指在片上缓冲区与计算阵列之间的搬运时间，由于在本文的加速器设计中采用是流水线方式设计，片上数据搬运时间会隐藏在网络计算当中，所以 $L_{on-chip}$ 为零。

3.3　片外访存延时

片外访存延时表示从 DDR 到片上缓存的读和写数据的时间,更具体是某种数据流下的总数据量 M^{dtflw} 乘以位宽 DW 除以总线带宽 BW，如式(10)所示。由于 FPGA 的片上容量有限，单层的数据可能超过片上缓冲区的大小，因此数据应该被分割成更小的块进行缓冲。在不同的数据流下，首先重用某种类型的数据,而其他类型的数据分块需要重复读取或写。所以每一种数据流都有不一样的访存数量。

$$L_{off-chip}=\frac{M^{dtflw}\times DW}{BW}=\sum_{layer}\frac{(M_{ifm}^{dtflw}+M_{wgt}^{dtflw}+M_{ofm}^{dtflw})\times DW}{BW} \tag{10}$$

4　加速器设计

为了验证延时模型的准确性和整个多层次协同设计的有效性，本文采用 Spinal HDL 这种新颖的硬件描述语言去设计了硬件的 CNN 加速器模板。Spinal HDL 类似 Verilog，采用硬件的思想，而不像高层次综合更多的考虑算法层面而忽略硬件设计的细节。由于在硬件端搜索加入了很多硬件参数，因此本文的 CNN 加速器模板实现了不同比特位、并行度参数化，从而可以使得整个模板可以通过不同的测试脚本用来测试不同的硬件参数组合。测试延时的获取是先获取对应子网络结构的配置指令，完成每一层的参数配置，再通过对加速器的控制模块状态机的读、写过程进行波形采样获取周期数。

整个加速器结构包含以下几部分：控制模块内部包含控制的状态机可以控制计算、读、写等过程，本文设计了不同的控制器来支持搜索空间中的不同数据流。访存模块是片内和片外访存的桥梁，可以将片上的数据写回 DDR，将 DDR 数据送到片上缓存。片上缓存模块实现数据和权值的缓存，分为两个数据 Buffer 和一个权值 Buffer。重排模块实现在缓存数据在输送到计算阵列之间的重新排布。计算阵列采用层间复用模式设计，每一层重复利用计算阵列，包含三个维度的并行(输入通道，卷积窗口和输出通道)以适配硬件端搜索的并行度设置，可以完成卷积、池化和矩阵相加等计算操作，如图 6 所示。

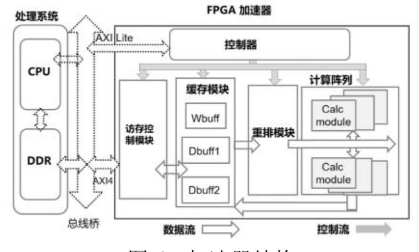


图 6　加速器结构

Fig. 6　The structure of accelerator

5　实验

5.1　实验设置

本文的实验环境配置如下：Ubuntu 16.04 LST 64 位系统，PyTorch1.71 深度学习框架, NVIDIA GTX 1080ti GPU。本文采

用的数据集是 MPII 数据集，包含 25K 张图像，采用 PCKh@0.5 评测指标。沙漏块堆叠数为 1，采用 RMS 作为网络训练时的优化器，学习率是 1e-3，超网络训练 220 轮，其中遗传算法搜索 20 次。利用的 FPGA 平台是 Xilinx Zynq 系列中的 7z045 作为资源约束。基于资源利用率过高会出现难以布线的情况的考虑，在搜索实验中，DSP 限制设置为 800 个，片上缓冲区总大小设置为1.312MB，两个数据Buffer 为512KB，权值Buffer 为288KB。

5.2 实验结果

a)传统加速方法。传统加速方法是基于固定算法结构，来考虑硬件设计的加速方法，因此本文针对原始沙漏网络结构进行数据流、并行度硬件参数和量化方式进行设计搜索。

图 7 是并行度与延时的关系，可以知道在相同数据流下，总并行度为 1536 的情况下延时最低,而并非在最大并行度 1600 的情况下，延时最低。这也体现出搜索的意义所在，并非并行度越高，其延时就越低，需要与具体卷积计算的通道数等契合。图 8 是数据流与延时的关系，可以知道在相同并行度下，针对于沙漏网络输出复用(OR)方式具有较低的延时。表 3 为不同量化方式的关键点检测结果，可以知道各种量化方式的每个关键点的检测结果，采用不同量化方式对原始沙漏网络结构量化后各个关节点的检测准确率有所下降，其中 Int8 损失最少，仅为 0.2。由于脚踝是比较容易被遮挡的部分，脚踝的检测效果最差，量化后其检测准确率下降最多。

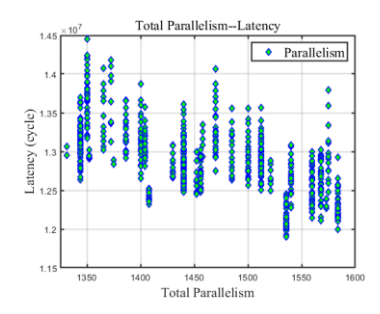


图 7　总并行度与延时的关系

Fig. 7　Relationship between total parallelism and latency

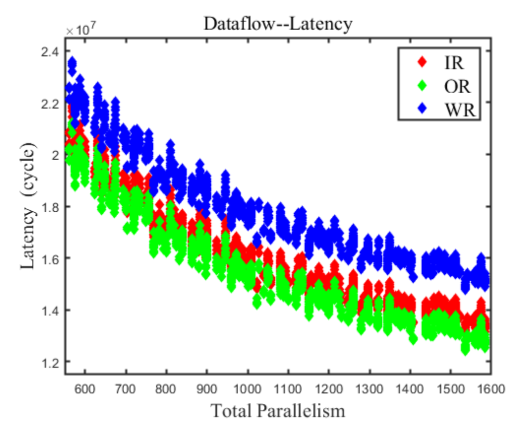


图 8　数据流与延时的关系

Fig. 8　Relationship between dataflow and latency

表 3　不同量化方式的关键点检测结果

Tab. 3　Key point estimation results of different quantization methods

方法	头	肩	手肘	手腕	臀部	膝盖	脚踝	平均
None	95.3	94.0	85.6	80.6	84.9	78.8	74.2	84.92
Int8	95.5	94.0	85.4	80.1	84.9	78.1	73.9	84.69
Fix8(1)	95.4	93.5	84.3	78.2	84.7	76.8	71.4	83.61
Fix8(2)	93.6	90.1	81.0	74.3	79.9	71.6	65.1	79.62
Fix8(3)	80.9	70.0	45.7	44.3	54.9	35.7	28.9	51.62

针对沙漏网络原始结构搜索结果的前五名，如表 4 所示，其中包括量化前、后的准确率、参数量以及数据流和并行度等。预测延时是指建立的延时模型获取的预测值，测试延时是指经过加速器模板仿真测试得到的延时，仿真测试主要是为了验证延时模型建模的正确性。经过多次重复测试得到，预测的时间延时和测试的时间延时平均误差只有 4.02%，足以表明延时模型的有效性。在表 4 可以知道，传统加速方法的搜索实验中，并行度组合为[Pox，Poy，Pic，Poc]=[8,3,8,8],数据流复用方式为输出复用 OR，量化方式为 Int8，延时最低，准确率损失最少，性能最优。

表 4　沙漏网络原始结构的搜索结果

Tab. 4　Search results of the original structure of hourglass network

排名	原始准确率	量化后准确率	量化方式	参数量/MB	复用方式	并行度组合	DSP 数目/个	预测延时/周期数	测试延时/周期数
1st	84.92	84.69	Int8	3.40	OR	[8, 3 ,8, 8]	768	1.250*10^7	1.284*10^7
2nd	84.92	84.69	Int8	3.40	OR	[8, 4, 8, 6]	768	1.256*10^7	1.290*10^7
3rd	84.92	84.69	Int8	3.40	OR	[6, 8, 8, 4]	768	1.257*10^7	1.291*10^7
4th	84.92	84.69	Int8	3.40	OR	[8, 8, 4, 6]	768	1.253*10^7	1.330*10^7
5th	84.92	84.69	Int8	3.40	OR	[8,12,4, 4]	768	1.281*10^7	1.359*10^7

b) 多层次协同搜索结果

针对沙漏模块重新构建了超网络进行训练，采用多层次协同搜索方法进行搜索，其搜索结果的前五名如下表 5 所示，表中包含沙漏块编码的网络结构、量化前、后的准确率、参数量、测试延时等。可以看到，最优的沙漏块结构为[0,1,1,0,1,1,2,0,1,0,0,1,0]，其量化后的准确率为 84.10，参数量为 2.23MB。由于在量化方式、数据流、并行度的搜索方面与传统方法一样采用遍历的方式进行，所以在这些参数的搜索基本一致。就平均而言，多层次搜索方法的原始准确率为 84.29，量化后准确率也有 84.23，损失很小，并且参数只有 2.27MB。至于测试延时的对比，会在后面进行。

c) 实验结果对比

由于本文工作是基于沙漏网络进行优化的，因此最直接的性能对比是针对原始结构，即文献[1]中的堆叠块为 1 时的结果以及传统针对原始结构的加速方法进行比较，如表 6 所示。

表 6 中的传统方法最优和平均分别是指表 4 的传统加速方法最优结果和前五名的平均结果，协同搜索最优和平均是指表 5 中的最优结果和前五名平均结果。就平均而言，本文方法比文献[1]堆叠块为 1 的结果减少了 83.3%的参数，准确率只下降了 0.69；比传统加速方法平均减少了 33.3%参数量，准确率只下降了 0.46，网络推理的测试总延时下降 22.1%，在沙漏块的测试延时下降了 67.8%。与其他姿态识别算法相比，本文的参数量是文献[7]的 0.83%，是文献[8]的 2%，是文献[9]的 0.9%。本文工作在准确率方面可能与优秀的算法存在一定的差距，但是在参数量和网络结构复杂度方面有一定的优势，更能适配到 FPGA 进行加速推理。

综上所述，本文的协同搜索方法在减少运行延时方面以很低的准确率损失，表现出比较好的性能。总体而言，多层次协同搜索比传统的设计过程具有更好的加速效果。

表 5 多层次协同搜索结果

Tab. 5 Search results of multi-level co-exploration

网络结构排名	原始准确率	量化后准确率	量化方式	复用方式	并行度组合	参数量/MB	整个网络延时/周期数	沙漏模块延时/周期数
[0,1,1,0,1,1,2,0,1,0,0,1,0]	84.21	84.10	Int8	OR	[8,3,8,8]	2.23	1.009*10^7	1.115*10^6
[0,0,1,0,1,0,2,0,1,1,0,0,0]	84.87	84.85	Int8	OR	[8,3,8,8]	2.50	1.050*10^7	1.541*10^6
[0,1,1,0,1,1,2,0,1,0,1,1,0]	83.96	83.92	Int8	OR	[8,3,8,8]	2.09	1.006*10^7	1.084*10^6
[0,0,1,1,0,0,2,1,0,0,0,1,0]	84.47	84.43	Int8	OR	[8,3,8,8]	2.37	1.036*10^7	1.385*10^6
[0,1,1,0,1,1,2,0,1,1,0,1,0]	83.92	83.87	Int8	OR	[8,3,8,8]	2.09	1.006*10^7	1.089*10^6
平均	84.29	84.23	Int8	OR	[8,3,8,8]	2.27	1.021*10^7	1.243*10^6

表 6 实验结果对比

Tab. 6 Comparison of experimental results

比较对象	参数量/MB	准确率	总延时/周期	沙漏块延时/周期
文献[1]	13.60	84.92	----	-----
传统方法最优	3.40	84.69	1.250*10^7	3.868*10^6
传统方法平均	3.40	84.69	1.311*10^7	4.046*10^6
协同搜索最优	2.23	84.10	1.009*10^7	1.115*10^6
协同搜索平均	2.27	84.23	1.021*10^7	1.243*10^6

6 结束语

本文针对姿态识别的沙漏网络，将沙漏块的结构、量化方式和硬件设计结合，以运行延时和准确率作为目标，建立了一个接近 FPGA 加速器真实运行的延时模型，提出了多层次协同搜索的方法，并经过设计的可配置的加速器模板验证其有效性。最终结果显示，这种协同设计方法在参数量和延时方面都优于传统的设计方法，并且只有很少的准确率下降。该方法对解决沙漏网络算法和硬件不匹配的问题有一定效果。

当然该工作仍有改进空间，例如考虑不同沙漏块的数目，卷积通道数等等，可以加入到搜索空间获取更高准确率的网络的同时保持更低的延时。

参考文献：

[1] Newell A, Yang K, Deng, J. Stacked hourglass networks for human pose estimation [C]// Proceedings of the 14th European Conference on Computer Vision, Springer, 2016: 483-499

[2] 周燕，刘紫琴，曾凡智，等. 深度学习的二维人体姿态估计综述 [J]. 计算机科学与探索, 2021, 15 (4): 641-657 (Zhou Yan, Liu Ziqin, Zeng Fanzhi, *et al.* Overview of 2D human pose estimation based on deep learning [J] Computer science and exploration, 2021, 15 (4): 641-657)

[3] 刘勇，李杰，张建林，等. 基于深度学习的二维人体姿态估计研究进展 [J]. 计算机工程, 2021, 47 (3): 1-16. (Liu Yong, Li Jie, Zhang Jianlin, *et al.* Research progress of 2D human pose estimation based on deep learning [J] Computer Engineering, 2021, 47 (3): 1-16.)

[4] Li Chuxi, Fan Xiaoya, Zhang Shengbing, *et al.* Hardware-Aware NAS Framework with Layer Adaptive Scheduling on Embedded System [C]// ASPDAC'21: 26th Asia and South Pacific Design Automation Conference. 2021.

[5] Zhen Dong, Gao Yizhao, Huang Qijing, *et al.* HAO: Hardware-aware neural Architecture Optimization for Efficient Inference [J]. 2021.

[6] Wei S E, Ramakrishna V, Kanade T, *et al.* Convolutional Pose Machines [J]. IEEE, 2016.

[7] Xiao Bin, Wu Haiping, Wei Yichen. Simple Baselines for Human Pose Estimation and Tracking [J]. arXiv e-prints, 2018.

[8] Sun Ke, Xiao Bin, Liu Dong, *et al.* Deep High-Resolution Representation Learning for Human Pose Estimation [J]. arXiv e-prints, 2019.

[9] Zhang Feng, Zhu Xiatian, Dai Hanbin, *et al.* Distribution-Aware Coordinate Representation for Human Pose Estimation [J]. 2019.

[10] Baker B, Gupta O, Naik N, *et al.* Designing Neural Network Architectures using Reinforcement Learning [J]. 2016.

[11] Zoph B, Vasudevan V, Shlens J, *et al.* Learning Transferable Architectures for Scalable Image Recognition [J]. 2017.

[12] Pham H, Guan M Y, Zoph B, *et al.* Efficient Neural Architecture Search via Parameter Sharing [J]. 2018.

[13] Cai Han, Yang Jiacheng, Han Song, *et al.* Path-Level Network Transformation for Efficient Architecture Search. In ICML, 2018. 3

[14] Brock A, Lim T, Ritchie J M, *et al.* SMASH: One-Shot Model Architecture Search through HyperNetworks [J]. 2017.

[15] Guo Zichao, Zhang Xiangyu, Mu Haoyuan, *et al.* Single Path One-Shot Neural Architecture Search with Uniform Sampling [J]. 2019.

[16] Courbariaux M, Hubara I, Soudry D, *et al.* Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to+1 or-1 [J]. 2016.

[17] Jacob B, Kligys S, Chen B, *et al.* Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference [J]. 2017.

[18] Wang Kuan, Liu Zhijian, Lin Yujun, *et al.* HAQ: Hardware-Aware Automated Quantization With Mixed Precision [C]// 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) . IEEE, 2020.

[19] Chan P, Park S, Park C S. Roofline-Model-Based Design Space Exploration for Dataflow Techniques of CNN Accelerators [J]. IEEE Access, 2020, 8: 172509-172523.

[20] Chen Y H, Krishna T, Emer J S, *et al.* Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks [J]. IEEE Journal of Solid-State Circuits, 2017, 52 (1): 127-138.

[21] Chen Y H, Yang T J, Emer J, *et al.* Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices [J]. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 9, no. 2, pp. 292-308, June 2019

[22] Ma Yufei, Cao Yu, Vrudhula S, *et al.* Optimizing the Convolution Operation to Accelerate Deep Neural Networks on FPGA, in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 7, pp. 1354-1367, July 2018

[23] Sharma H, Park J, Suda N, *et al.* Bit Fusion: Bit-Level Dynamically Composable Architecture for Accelerating Deep Neural Networks [J]. 2017.

[24] Gong Chengyue, Jiang Zixuan, Wang Dilin, *et al.* Mixed Precision Neural Architecture Search for Energy Efficient Deep Learning [C]// ICCAD 2019. pp. 1-7

[25] Jiang Weiwen, Li Yang, Sha H M, *et al.* Hardware/Software Co-Exploration of Neural Architectures [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, PP (99): 1-1.

[26] Luo Xiangzhong, Liu Di, Huai Shuo, *et al.* HSCoNAS: Hardware-Software Co-Design of Efficient DNNs via Neural Architecture Search [J]. In DATE, 2021.

[27] Jiang Yuhang, Wang Xin, Zhu Wenwu. Hardware-Aware Transformable

Architecture Search with Efficient Search Space [C]// 2020 IEEE International Conference on Multimedia and Expo (ICME) . IEEE, 2020.

[28] Qiu Zhongyu, Li Jianli, Liang Dongbao, *et al.* A CNN/FPGA Co-Exploration Method Based on Hourglass Network. [C]// 2021 IEEE 4th International Conference on Electronics Technology (ICET) , 2021, pp. 333-337

[29] Jiang Weiwen, Zhang Xinyi, Sha H M, *et al.* Accuracy vs. Efficiency: Achieving Both through FPGA-Implementation Aware Neural Architecture Search. [C]// 2019 56th ACM/IEEE Design Automation Conference (DAC) , 2019, pp. 1-6.

[30] Jiang Weiwen, Lou Qiuwen, Yan Zheyu, *et al.* Device-Circuit-Architecture Co-Exploration for Computing-in-Memory Neural Accelerators [J]. IEEE Transactions on Computers, 2020, PP (99): 1-1.